UDC 004.05 **DOI: 10.20998/2079-0023.2020.01.20**

V. I. ZYBIN, I. V. LIUTENKO

DESIGNING INFORMATION SUPPORT FOR EVALUATING THE QUALITY OF EMBEDDED SOFTWARE

This article presents a system for evaluating the quality of embedded software using a decision system based on fuzzy logic. These approaches will improve the assessment of software quality, due to its features. This article defines the main criteria for software quality used in assessing the quality of the software. The main literature was examined, in which fuzzy logic was described, decision-making systems using fuzzy logic, as well as software quality assessment systems, including software for embedded systems. The main characteristics and properties of embedded systems were considered. Based on the considered characteristics and properties of embedded systems, the ranking of criteria was made, which will be further used in the software quality assessment methodology. The main criteria that are used to evaluate the quality of software were considered, and the criteria presented were distributed according to the degree of influence on the assessment of the quality of software of embedded systems. Fuzzy logic was considered, and more precisely: the basic properties of fuzzy logic and fuzzy numbers, the basic mathematical operators applied to fuzzy numbers. The system for constructing rules for the rule base, as well as the defuzzification process, built on the basis of the centroid method, is analyzed. An example of software evaluation for embedded systems was considered. In this example, linguistic variables were determined, as well as their numerical ranges, which were used for the initial assessment of the quality criteria of this software. Each range of ratings was distributed according to the influence of a criterion on software quality. The output linguistic variable and its numerical value were also determined. In the end, based on the set values, an estimate of the set software was derived. The theoretical result obtained in this article is the basis for constructing a system for evaluating software quality for embedded systems.

Keywords: decision-making, fuzzy logic, embedded systems, software quality, software assessments, software testing.

В. І. ЗИБІН, І. В. ЛЮТЕНКО

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОЦІНКИ ЯКОСТІ ПЗ ВБУДОВАНИХ СИСТЕМ

У даній статті представлена система для оцінки якості програмного забезпечення вбудованих систем з використанням системи прийняття рішень на основі нечіткої логіки. Дані підхід дозволить поліпшити оцінку якості програмного забезпечення, за рахунок урахування його особливостей. У даній статті визначено основні критерії якості програмного забезпечення, використовувані при оцінці якості даного програмного забезпечення. Була оглянута основна література, в якій була описана нечітка логіка, системи прийняття рішень, що використовують нечітку логіку, а також системи оцінки якості програмного забезпечення, в тому числі і програмного забезпечення для вбудованих систем. Були розглянуті основні характеристики та властивості вбудованих систем. На підставі розглянутих характеристик і властивостей вбудованих систем виробилося ранжування критеріїв, які в подальшому будуть використовуватися в методиці оцінки якості програмного забезпечення. Були розглянуті основні критерії, які використовуються для оцінки якості програмного забезпечення, а також представлені критерії, які були розподілені за ступенем впливу на оцінку якості програмного забезпечення вбудованих систем. Була розглянута нечітка логіка, а точніше: основні властивості нечіткої логіки і нечітких чисел, основні математичні оператори, що застосовуються до нечітким числах. Розібрана система побудови правил для бази правил, а також процес дефазифікації, побудований на підставі центоїдного методу. Було розглянуто приклад оцінки програмного забезпечення для вбудованих систем. В даному прикладі були визначені лінгвістичні змінні, а також їх числові діапазони, які використовувалися для первісної оцінки критеріїв якості даного програмного забезпечення. Кожен діапазон оцінок був розподілений згідно впливу критерію на якість програмного забезпечення. Також була визначена вихідна лінгвістична змінна і її числове значення. В кінці, на основі заданих значень була виведена оцінка заданого програмного забезпечення. Отриманий теоретичний результат в даній статті є основою для побудови системи для оцінки якості програмного забезпечення для вбудованих системи.

Ключові слова: прийняття рішень, нечітка логіка, вбудовані системи, якість програмного забезпечення, оцінка програмного забезпечення, тестування програмного забезпечення.

В. И. ЗЫБИН, И. В. ЛЮТЕНКО

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОЦЕНКИ КАЧЕСТВА ПО ВСТРАИВАЕМЫХ СИСТЕМ

В данной статье представлена система для оценки качества программного обеспечения встраиваемых систем с использованием системы принятия решений на основе нечёткой логики. Данные подход позволит улучшить оценку качества программного обеспечения, за счёт учёта его особенностей. В данной статье определены основные критерии качества программного обеспечения, используемые при оценке качества данного программного обеспечения. Была осмотрена основная литература, в которой была описана нечёткая логика, системы принятия решений, использующие нечёткую логику, а также системы оценки качества программного обеспечения, в том числе и программного обеспечения для встраиваемых систем. Были рассмотрены основные характеристики и свойства встраиваемых систем. На основании рассмотренных характеристик и свойств встраиваемых систем произвелось ранжирование критериев, которые в дальнейшем будут использоваться в методике оценки качества программного обеспечения. Были рассмотрены основные критерии, которые используются для оценки качества программного обеспечения, а также представленные критерии были распределены по степени влияния на оценку качества программного обеспечения встраиваемых систем. Была рассмотрена нечёткая логика, а точнее: основные свойства нечёткой логики и нечётких чисел, основные математические операторы, применяемые к нечётким числам. Разобрана система построения правил для базы правил, а также процесс дефазификации, построенный на основании метода центроидов. Был рассмотрен пример оценки программного обеспечения для встраиваемых систем. В данном примере были определены лингвистические переменные, а также их числовые диапазоны, которые использовались для первоначальной оценки критериев качества данного программного обеспечения. Каждый диапазон оценок был распределён согласно вилянию критериев на качество программного обеспечения. Также был определена выходная лингвистическая переменная и её числовое значение. В конце, на основе заданных значений была выведена оценка заданного программного обеспечения. Полученный теоретический результат в данной статье является основой для построения системы для оценки качества программного обеспечения для встраиваемых системы.

Ключевые слова: принятие решений, нечёткая логика, встраиваемые системы, качество программного обеспечения, оценка программного обеспечения, тестирование программного обеспечения.

Introduction. Test automation solution – is a realization (implementation) of a test automation architecture, i.e., a combination of components implementing a specific test automation assignment. According to ISO 25010, quality software meets the following criteria [1]:

- Functional Suitability;
- Performance efficiency;
- Compatibility;
- Usability;
- Reliability;
- Security;
- Maintainability;
- Portability.

Embedded systems are used everywhere, including in such areas as medicine, astronautics, etc. In this regard, the quality of embedded software is very important. After all, poor-quality software can cause huge losses.

Due to the growth of device functionality and, accordingly, the growth of software code sizes, quality is becoming an increasingly urgent problem. The literature uses many different ways to evaluate software quality. The main problem of all the proposed solutions is that they do not take into account the specific features of embedded software (for example, a small amount of memory in devices). To solve the problem of software quality, it is proposed to use fuzzy logic. The advantage of using fuzzy logic is that this approach will allow you to take into account the features of this type of software.

In this article, to solve the problem, we use literature describing fuzzy logic and the ISO 25010 standard to determine the main criteria for software quality. Zade [2] was one of the first to describe the basics of fuzzy logic. Gieseckea [3] examined various architectural constraints that should have served as the basis for software quality assessment. Siavvas [4] created the QATCH framework, which allows you to evaluate software quality based on customer requirements. Pasrija [5] used the choquet integral Approach to evaluate software quality. Ahrem [6] illustrates an example of the use of fuzzy logic in the decision-making system. Gorbachenko [7] is considering an ISO standard to create a software testing system. In the work of Klyuyev [8], fuzzy logic is used for a general assessment of the quality of software.

The advantage of fuzzy logic is that it allows you to take into account the features of a particular type of software. Such an approach has the advantage over others that increases the accuracy of the assessment.

The aim of this work is to develop a method for testing embedded software based on fuzzy logic.

This article consists of such sections. Introduction – where basic information is presented. In literature review discusses the main sources that was used when writing the article. Methods, the methodology of building a system is considered in this section. The results section discusses the results for the methods section. The conclusions section contains conclusions regarding to this work.

Literature review. In an age when embedded systems are becoming an increasingly large part of our lives, its quality should never be lower than ever. Now there are many new approaches to assessing the quality of software.

Zade [2] is one of the first to describe fuzzy logic.

Gieseckea [3] considered various architectural constraints that can be used for reuse and to improve the quality of software. He proposed to use two classes of architectural restrictions: Pattern-based concepts and Style-based concepts. The disadvantage of this approach is that it is always applicable. Since there are many programming languages and types of software, these architectures are not always possible to implement.

Siavvas [4] in his work proposed an adaptive framework for assessing the quality of the software QATCH (Quality Assessment Tool CHain). Based on the criteria of the ISO 25010 standard, this framework allows software evaluation. The disadvantage of this approach is that it is embedded in the program code. This approach does not take into account the limited resources of embedded systems and cannot be effectively used in such systems.

Pasrija [5] in his work suggested using Choquet Integral to evaluate software quality. This approach uses fuzzy numbers. But in this paper, a generalized example is used that does not take into account the features of each software.

The work of Ahrem [6] describes examples of the use of fuzzy logic in the decision-making system.

Gorbachenko [7] described the criteria and methodology for assessing software quality.

Klyuyev [8] uses fuzzy logic to evaluate software quality.

Garusi [9] gave a full review of the literature on evaluating the quality of embedded software. In his work, Google scholar and scopus were used as the main source of articles. As a result, the sample articles were classified by this type:

1. Test-case design;
3. Test scripting;
4. Test execution;
5. Test evaluation;
6. Test-result reporting;
7. Test automation;
8. Test management;
9. Other test engineering activities;

As a result, five of the most cited articles were highlighted.

The Minhas [10] in his article uses Regression testing to determine the quality of embedded software. The main idea of this approach is that it does not value the software of embedded systems itself, but rather shows the influence of the new functionality on the quality of the system.

Seo [11] in their article developed a system for evaluating the performance of embedded system software based on a kernel hack. This system is only capable of evaluating performance; therefore, it is not inconvenient to show possible errors in the software of the embedded system and other problems. An additional disadvantage of this system is that it requires output to connect the system and evaluate it.

Burakov [12] described the criteria and methodology for assessing software quality. The downside of this work is that some criteria are not suitable for embedded systems.

The work of and Pronina [13] describes examples of the use of fuzzy logic in the decision-making system.

Rudkovska [14] and Grinyaev [15] describe additional information about fuzzy sets. Those works describes the basic laws of fuzzy logic.

This article uses ISO 25010, on the basis of which linguistic variables are taken. The disadvantage of this approach is that it does not take into account the features of the programming language and the type of system.

Methods. The quality of a system is the degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides the value. Those stakeholders' needs are precisely what is represented in the quality model, which categorizes the product quality into characteristics and sub-characteristics. To test the quality of testing, special quality criteria are used. One of the sources of such criteria is the ISO25010 standard. ISO/IEC 25010 comprises the eight quality characteristics [1] shown in the following fig. 1.

According to the ISO standard, these criteria have the following meaning:

Functional Suitability. This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics: functional completeness, functional correctness and functional appropriateness.

Performance Efficiency. This characteristic represents the performance relative to the amount of resources used under stated conditions. This characteristic is composed of the following sub-characteristics: time behavior, resource utilization and capacity.

Compatibility. Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions while sharing the same hardware or software environment. This characteristic is composed of the following sub-characteristics: co-existence, interoperability.

Usability. Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This characteristic is composed of the following sub-characteristics: appropriateness recognizability, learnability, operability, user error protection, user interface aesthetics and accessibility.

Reliability. Degree to which a system, product or component performs specified functions under specified conditions for a specified period. This characteristic is composed of the following sub-characteristics: maturity, availability, fault tolerance, recoverability.

Security. Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization. This characteristic is composed of the following sub-

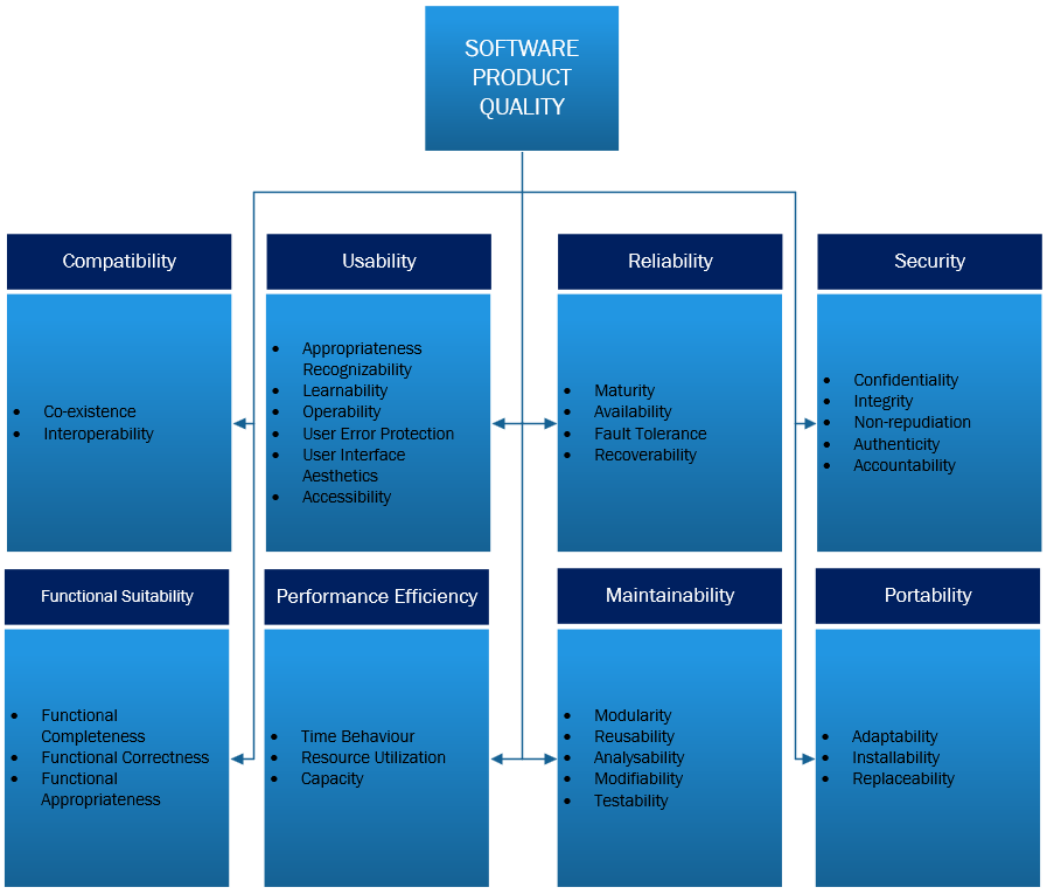

Fig. 1. Software quality criteria

characteristics: confidentiality, integrity, nonrepudiation, accountability, authenticity.

Maintainability. This characteristic represents the degree of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements. This characteristic is composed of the following sub-characteristics: modularity, reusability, analysability, modifiability, testability.

Portability. Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics: adaptability, installability, replaceability.

For a more effective assessment of software quality, each criterion should be divided according to the degree of influence. To separate the criteria, the basic properties of the software of embedded systems should be determined.

An embedded system is a controller programmed and controlled by a real-time operating system (RTOS) using a special function in a larger mechanical or electrical system.

The main features of the embedded systems are:

- real-time work (almost always);
- various, often difficult, operating conditions;
- autonomy of work (lack of operator, power restrictions);
- high requirements for reliability and safety of operation;
- limited resources;
- critical applications (Dependable Applications) related to human health and life.

Based on the features of embedded systems, quality criteria can be divided into the following categories:

- Criteria of high importance: Performance Efficiency, Usability, Reliability;
- Criteria of medium importance: Functional Suitability, Maintainability;
- Criteria of low importance: Compatibility, Security, Portability.

Fuzzy logic is used to evaluate the criteria. This approach is able to take into account how much each of the criteria is significant for determining the quality of software in embedded systems.

A fuzzy set is a class of objects with a continuum of grades of membership. Such a set characterized by a membership (characteristic) function that assigns to each object a grade of membership ranging between zero and one. The notions of inclusion, union, intersection, complement, relation, convexity, etc., extended to such sets, and various properties of these notions in the context of fuzzy sets are established. In particular, a separation theorem for convex fuzzy sets proved without requiring that the fuzzy sets be disjoint. L. A. Zadeh represented fuzzy sets algorithm in 1965. In that work for mathematical definition use X, which a space point (objects), with a generic element of X denoted by x. Thus, $X = \{x\}$. A fuzzy set (class) A in X is characterized by a membership (characteristic) function $f_A(x)$ which associates with each point 2 in X a real number in the interval [0, 1], with the value of $f_A(x)$ at x representing the "grade of membership" of x in A. Thus, the nearer the value of $f_A(x)$ to unity, the higher the grade of membership of x in A. When A is a set in the ordinary sense of the term, its membership function can take oil only two values 0 and 1, with $f_A(x) = 1$ or 0 according as x does or does not belong to A. Thus, in this case $f_A(x)$ reduces to the familiar characteristic function of a set A [2].

In the theory of fuzzy systems, fuzzy sets are those that are on the axis of real numbers. Fuzzy number can determine the set A, which is on the set of real numbers $A \subseteq R$, the membership function of which has the value $\mu_A: R[0, 1]$ and meets the conditions:

1. $\sup_{x \in R} \mu_A(x) = 1$, in other way the fuzzy set A is normalized;
2. $\mu_A[\lambda k_1 + (1 - \lambda)x \min\{\mu_A(x_1), \mu_A(x_2)\}]$, in other way the fuzzy set A is convex;
3. $\mu_A(x)$.

Fuzzy sets can be divided into positive and negative. A fuzzy number is positive when $\mu_A(x) = 0$ for $x < 0$ all, negative when $\mu_A(x) = 0$ for all $x > 0$ [14].

Fuzzy numbers have their own binary operations, which are defined through generalizations of operations for clear numbers. These operators meet the conditions:

- addition $\mu_{A+B}(x) = \max\limits_{z = x + y} (\mu_A(x) \wedge \mu_B(y))$; $\forall x, y, z \in R$;
- subtraction $\mu_{A-B}(x) = \max\limits_{z = x - y} (\mu_A(x) \wedge \mu_B(y))$; $\forall x, y, z \in R$;
- multiplication $\mu_{A \cdot B}(x) = \max\limits_{z = x \cdot y} (\mu_A(x) \wedge \mu_B(y))$; $\forall x, y, z \in R$;
- division $\mu_{A/B}(x) = \max\limits_{z = \frac{x}{y}, y \neq 0} (\mu_A(x) \wedge \mu_B(y))$; $\forall x, y, z \in R$.

These algebraic operations have a very large volume of calculations, so often fuzzy numbers are represented in LR form, where L is the left part of the number, R is the right part of the number. The fuzzy number in LR form has form:

$$\begin{aligned} &L\left(\frac{m - x}{\alpha}\right); \alpha > 0; \forall x \leq m, \\ &R\left(\frac{m - x}{\beta}\right); \beta > 0; \forall x \leq m. \end{aligned} \quad (1)$$

Where L and R are functions that have the properties:

$$\begin{aligned} L(-x) &= L(x), \\ L(0) &= 1. \end{aligned} \quad (2)$$

The function L decreases monotonically on the interval $[0, +\infty]$. m is the mean value of the fuzzy number, α is the deviation from the mean value on the left, β is the deviation of the value on the right. If $\alpha = \beta = 0$, then the fuzzy number A becomes clear. Thus, fuzzy numbers in LR form can be represented as $A = \{m_A, \alpha_A, \beta_A\}$, and the operations have the form:

- addition operation: $A + B = (m_A, \alpha_A, \beta_A) + (m_B, \alpha_B, \beta_B) = (m_A + m_B, \alpha_A + \alpha_B, \beta_A + \beta_B)$;
- subtraction operation: $A - B = (m_A, \alpha_A, \beta_A) - (m_B, \alpha_B, \beta_B) = (m_A - m_B, \alpha_A - \alpha_B, \beta_A - \beta_B)$;
- multiplication operation: $A \cdot B = (m_A, \alpha_A, \beta_A) \cdot (m_B, \alpha_B, \beta_B) = (m_A \cdot m_B, m_B\alpha_A + m_A\alpha_B, m_B\beta_A + m_A\beta_B)$ [15];

To solve the problem of choosing an automated testing system, using a fuzzy inference, which determines the non-linear mapping of the input data vector into a scalar output value using fuzzy rules. A fuzzy logic output with a multidimensional output considered as a set of independent fuzzy logic outputs with a multidimensional input and a one-dimensional output.

A defuzzifier maps a fuzzy output set to a fuzzy set containing a range of output values. Defuzzifier converts to a single numerical value, convenient for further use. There are several methods of defuzzification: centroid, maximum and maximum centeroid method. In this paper, the centroid method is used.

In the centroid method, the center of gravity (centroid) is determined, which is the result of $\bar{y}$. For continuously and discretely defined values of fuzzy numbers of a set, respectively:

$$\bar{y} = \frac{\int_b^a y\mu(y)dy}{\int_b^a \mu(y)dy}, \bar{y} = \frac{\sum_{i=1}^n y_i\mu(y_i)}{\sum_{i=1}^n \mu(y_i)}. \quad (6)$$

The scheme of the mechanism of logical conclusion is shown in fig. 2.

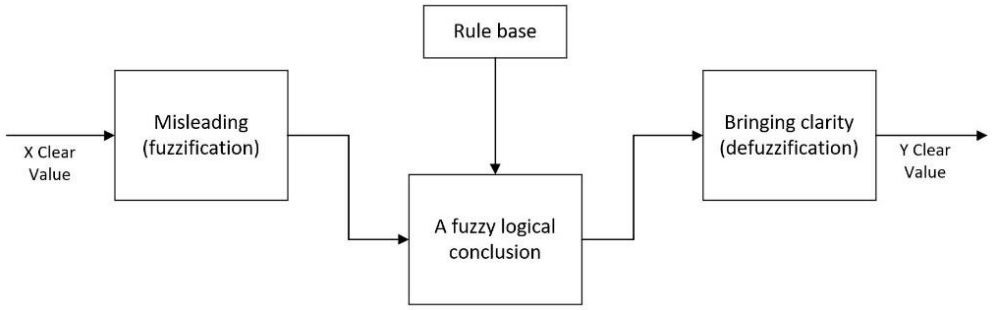

Fig. 2. Mechanism of logical conclusion

Fuzzy inference consists of three components: a fuzzifier, a logical inference mechanism, and a defuzzifier. The fuzzifier determines the degree to which input values belong to fuzzy input sets – linguistic variables.

The core of the inference mechanism is a rule base containing linguistic rules derived from static numeric data. Rule base consists of a set of rules in the format:

$$\begin{aligned} Rule \ = \ & \text{if } x_1 \text{ has } T_{x_1}^{any} \\ & \text{and if } x_2 \text{ has } T_{x_2}^{any} \ldots \\ & \text{and if } x_m \text{ has } T_{x_m}^{any} \\ & \text{then } y \text{ has } T_y^{any}, \end{aligned} \quad (3)$$

where x_1,x_2,x_m is linguistic variables, $T_{x_1}^{any}$, $T_{x_2}^{any}$, $T_{x_m}^{any}$, T_y^{any} is rules, y is linguistic output.

If more than one condition is used, then it is necessary to use a fuzzy operator to determine the result of applying this rule, that is, determine the degree of membership for a fuzzy set of outputs. To do this, use the minimum operators (3) and products (4):

$$\mu(y) = \min\big(\mu(x_1), \mu(x_2), \ldots, \mu(x_m)\big), \quad (4)$$

$$\mu(y) = \mu(x_1) \cdot \mu(x_2) \cdot \ldots \cdot \mu(x_m), \quad (5)$$

where $\mu(x_1) \cdot \mu(x_2) \cdot \ldots \cdot \mu(x_m)$ – degree to which the input values and the application result belong to the corresponding fuzzy sets of linguistic variables.

Results. An example of software evaluation is given as a result. To start the assessment, linguistic variables and their value of a fuzzy number should be determined.

Each of the criteria has such linguistic variables:

x_n = {bad, normal, good}, where $n = 1, 2, \ldots, n$, where n is total number of criteria.

Each criterion has its own range of values depending on the importance of this criterion. The ranges of values are presented in table 1.

Table 1 – Range of values for linguistic variables in different group

	Bad	Normal	Good
Criteria of high importance	[1, 4]	[5, 8]	[9, 10]
Criteria of medium importance	[1, 3]	[4, 7]	[8, 10]
Criteria of low importance	[1, 2]	[3, 6]	[7, 10]

The assessments of experts according to table 1 are converted from numerical variables to linguistic ones. Based on linguistic variables, a rule base is constructed according to formula (5). Based on these rules, a final grade is derived. The final grade is also a linguistic variable and has its own meanings: y = {bad quality, low quality, normal quality, above normal quality, high quality}. These linguistic variables have the following numerical values:

bad quality – 2, low quality – 4, normal quality – 6, above normal quality – 8, high quality – 10. For example, software that has such ratings:

- Functional Suitability – 8;
- Performance Efficiency – 6;
- Compatibility – 2;
- Usability – 6;
- Reliability – 8;
- Security – 9;
- Maintainability – 6;
- Portability – 4.

According to Table 1, these criteria have the following linguistic variables:

- Functional Suitability – Good;
- Performance Efficiency – Normal;
- Compatibility – Bad;
- Usability – Normal;
- Reliability – Normal;
- Security – Good;
- Maintainability – Normal;
- Portability – Normal.

According to the rule base, based on these values, the output score of this software is equal to: above average or in a garble value of 8.

Conclusions. This article describes the quality criteria and their impact on evaluating the quality of software for embedded systems, the basics of fuzzy logic and at the end describes an example of software evaluation. The described approach is noteworthy in that it allows to get an assessment of the quality of software based on its features, which allows you to improve the assessment of software quality.

This approach can be used not only to assess software quality. For example, in article [13], a system for spinning a solution based on fuzzy logic is used.

As a future work, writing a framework that, based on this issue, will evaluate the system.

References

1. *ISO/IEC 25010*. URL: http://iso25000.com/index.php/en/iso-25000-standards/iso-25010 (accessed 27.04.2020).
2. Zadeh L. A. Fuzzy Sets. *Information and control*. 1965. No. 8. P. 338–353.
3. Gieseckea S., Hasselbringa W., Riebischb M. Classifying architectural constraints as a basis for software quality assessment. *Advanced Engineering Informatics*. 2007. Vol. 21, issue 2. P. 169–179.
4. Siavvas M. G., Chatzidimitriou K. C., Symeonidis A. L. QATCH - An adaptive framework for software product quality assessment. *Expert Systems with Applications*. 2017. Vol. 86. P. 350–366.
5. Pasrija V., Kumar S., Srivastava P. R. Assessment of Software Quality: Choquet Integral Approach. *Procedia Technology*. 2012. Vol. 6. P. 153–162.
6. Ахрем А. А., Ашинянц М. Р., Петров С. А. Нечеткий логический вывод в системе принятия решений. *Труды ИСА РАН*. 2007. Т. 29. С. 265–275.
7. Горбаченко И. М. Оценка качества программного обеспечения для создания систем тестирования. *Фундаментальные исследования* 2013. № 6 (часть 4). С. 823–827.
8. Клюев Е.И., Гриненко Е.А. Подход к оценке качества программных средств. *Інженерія програмного забезпечення* 2014. № 3 (19). С. 5–14.
9. Garousia V., Feldererbe M., Karapıçakc Ç. M., Yılmazd U. Testing embedded software: A survey of the literature. *Information and Software Technology* 2018. Vol. 104. P. 14–45.
10. Mehmood N., Petersen M. K., Börstler J., Wnuk K. Regression testing for large-scale embedded software development – Exploring the state of practice. *Information and Software Technology*. 2020. Vol. 120. Article 106243.
11. Seo J., Choi B., Yang S. Lightweight embedded software performance analysis method by kernel hack and its industrial field study. *Journal of Systems and Software*. 2012. Vol. 85, issue 1. P. 28–42.
12. Бураков В.В. Методика оценки качества программных средств. Известия высших учебных заведений. Приборостроение. 2008. Т. 51, № 1. С. 35–41.
13. Пронина О. И., Пятикоп Е. Е. Использование нечетких множеств при определении класса автомобиля. *Вісник Нац. техн. ун-ту «ХПІ»: зб. наук. пр. Сер.: Системний аналіз, управління та інформаційні технології*. Харків: НТУ «ХПІ», 2017. № 28 (1250). С. 41–48.
14. Рутковская Д., Пилиньский М., Рутковский Л. *Нейронные сети, генетические алгоритмы и нечеткие системы*. Москва: Горячая линия - Телеком, 2006. 452 с.
15. Гриняев Ю. В. *Теория нечетких множеств. Учебное пособие для студентов*. Томск: Томский государственный университет систем управления и радиоэлектроники, 2008. – 141 с.

References (transliterated)

1. *ISO/IEC 25010*. Available at: http://iso25000.com/index.php/en/iso-25000- standards/iso-25010 (accessed 27.04.2020).
2. Zadeh L. A. Fuzzy Sets. *Information and control* 1965, vol. 8, pp. 338–353.
3. Gieseckea S., Hasselbringa W., Riebischb M. Classifying architectural constraints as a basis for software quality assessment. *Advanced Engineering Informatics*. 2007, vol. 21, issue 2, pp. 169–179.
4. Siavvas M. G., Chatzidimitriou K. C., Symeonidis A. L. QATCH - An adaptive framework for software product quality assessment. *Expert Systems with Applications*. 2017, vol. 86, pp. 350–366.
5. Pasrija V., Kumar S., Srivastava P. R. Assessment of Software Quality: Choquet Integral Approach. *Procedia Technology*. 2012, vol. 6, pp. 153–162.
6. Ahrem A. A., Ashinyants M. R., Petrov S. A. Nechetkij logicheskij vyvod v sisteme prinyatiya reshenij [Fuzzy inference in the decision-making system]. *Trudy ISA RAN* [Proceedings of ISA RAN]. 2007, vol. 29, pp. 265–275.
7. Gorbachenko I. M. Ocenka kachestva programmnogo obespecheniya dlya sozdaniya sistem testirovaniya [Quality assessment of software for creating testing systems]. *Fundamental'nye issledovaniya* [Basic research]. 2013, no. 6 (part 4), pp 823–827.
8. Klyuev E.I., Grinenko E.A. Podhod k ocenke kachestva programmnyh sredstv [Approach to software quality assessment]. *Inzheneriia prohramnoho zabespechennia* [Software Security Engineering]. 2014, no. 3 (19), pp. 5–14.
9. Garousia V., Feldererbe M., Karapıçakc Ç. M., Yılmazd U. Testing embedded software: A survey of the literature. *Information and Software Technology*. 2018, vol. 104, pp. 14–45.
10. Mehmood N., Petersen M. K., Börstler J., Wnuk K. Regression testing for large-scale embedded software development – Exploring the state of practice. *Information and Software Technology*. 2020, vol. 120, article 106243.
11. Seo J., Choi B., Yang S. Lightweight embedded software performance analysis method by kernel hack and its industrial field study. *Journal of Systems and Software*. 2012, vol. 85, issue 1, pp. 28–42.
12. Burakov V. V. Metodika ocenki kachestva programmnyh sredstv [Software Quality Assessment Methodology]. *Izvestiya vysshih uchebnyh zavedenij. Priborostroenie* [News of higher educational institutions. Instrumentation]. 2008, vol. 51, no. 1, pp. 35–41..
13. Pronina O. I., Patykop E. E. Ispol'zovanie nechetkih mnozhestv pri opredelenii klassa avtomobilya [Using fuzzy sets when determining a car class]. *Visnyk Natsionalnoho tekhnichnoho universytetu «KhPI»: zbirnyk naukovykh prats. Seriia: Systemnyi analiz, upravlinnia ta informatsiini tekhnolohii* [Bulletin of the National Technical University "KhPI": a collection of scientific papers. Series: Systems Analysis, Control and Information Technology]. 2017, no. 28 (1250), pp. 41–48.
14. Rutkovska D., Pilinski M., Rutkovski L. Sieci neuronowe, algorytmy genetyczne i systemy rozmyte. Warszawa, Łodź, Wydawnictwo Naukowe PWN, 2004. 410 s. (Russ. ed.: Rutkovska D., Pilinski M.,

Rutkovski L. *Nejronnye seti, geneticheskie algoritmy i nechetkie sistemy*. Moscow, Goryachaya liniya - Telekom Publ., 2006. 452 p.).
15. Grinyaev Yu. V. *Teoriya nechetkih mnozhestv. Uchebnoe posobie dlya studentov* [Theory of fuzzy sets. Study guide for students]. Tomsk, Tomskij gosudarstvennyj universitet sistem upravleniya i radioelektroniki Publ., 2008. 141 p.

Received 04.05.2020

Відомості про авторів / Сведения об авторах / About the Authors

Зибін Владислав Іванович – Національний технічний університет «Харківський політехнічний інститут», студент кафедри Програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-6299-3552; e-mail: vladyslavzybin2@gmail.com

Лютенко Ірина Вікторівна – кандидат технічних наук, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри програмної інженерії та інформаційних технологій управління; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: cherliv68@gmail.com

Зыбин Владислав Иванович – Национальный технический университет «Харьковский политехнический институт»,студент кафедры Программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0001-6299-3552; e-mail: vladyslavzybin2@gmail.com

Лютенко Ирина Викторовна – кандидат технических наук, Национальный технический университет «Харьковский политехнический институт», доцент кафедры программной инженерии и информационных технологий управления; г. Харьков, Украина; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: cherliv68@gmail.com

Zybin Vladyslav Ivanovich National Technical University «Kharkiv Polytechnic Institute», student of the Department of Software Engineering and Information Technology Management; Kharkiv city, Ukraine; ORCID: https://orcid.org/0000-0001-6299-3552; e-mail: vladyslavzybin2@gmail.com

Liutenko Iryna Victorivna – Candidate of Engineering Sciences, National Technical University "Kharkiv Polytechnic Institute", Associate Professor, Department of Software Engineering and Management Information Technology; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-4357-1826; e-mail: cherliv68@gmail.com

ЗМІСТ